Flexible Flight Software Facilitating Evolving Devices and Interfaces

May 2, 2017

V4e

Douglas Forman
Intelligent Systems Division
NASA Ames Research Center

Outline

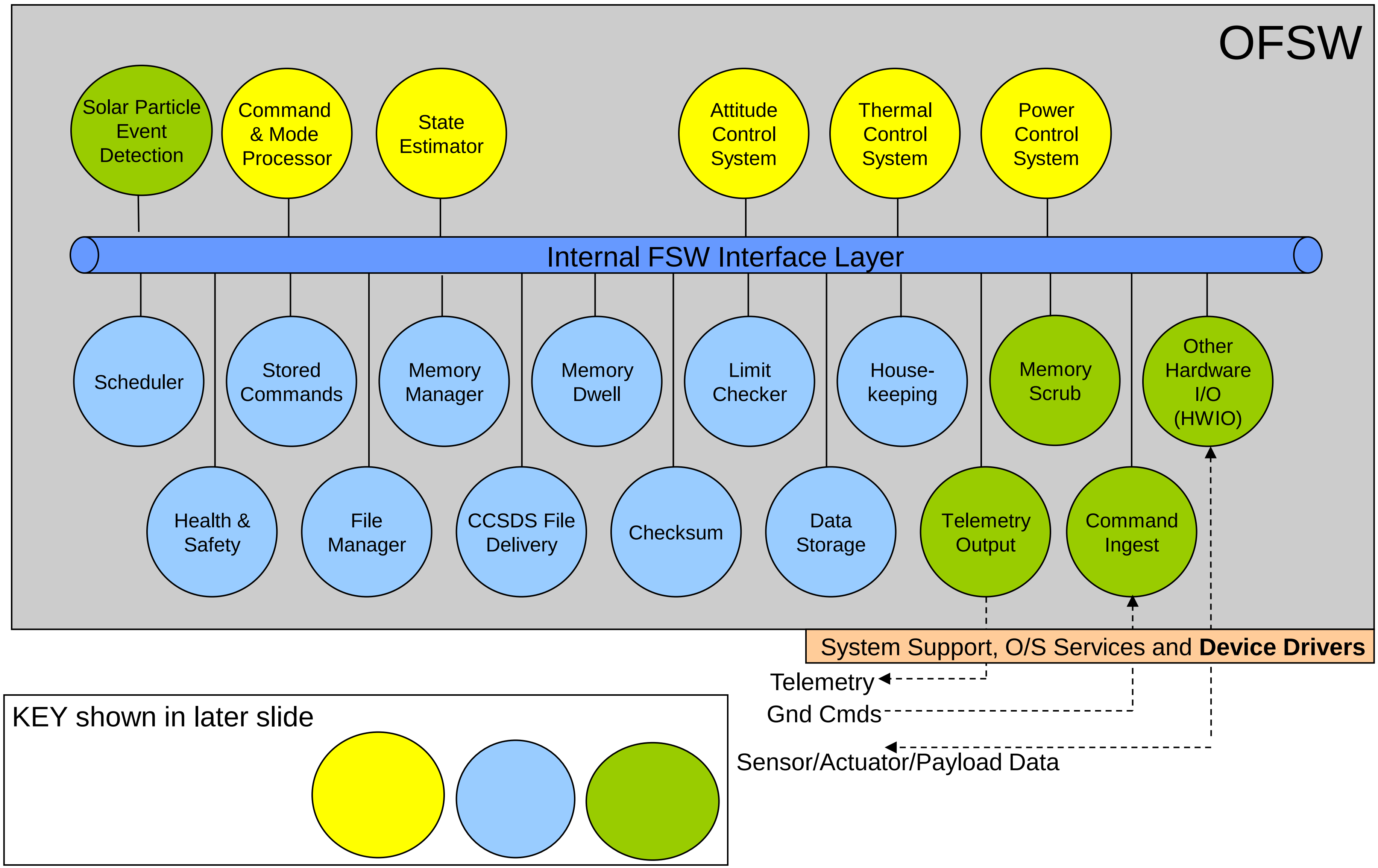
- **BioSentinel FSW Overview**
- **BioSentinel H/W Availability Challenge**
- **NASA Ames FSW Infrastructure**
- **BioSentinel Specific Solutions**
  - Iris Transponder FSW Evolution
  - C&DH Interface eXtension Card (IXC) Spacewire-to-Serial FSW Evolution

BioSentinel FSW Overview

BioSentinel FSW Driving Requirements

- **Payload/Science Maintenance**
  - PLRD-029 Science Data, SRD-035 IO Design, PLRD-024 SPE Detection, PLRD-025 SPE Response
- **Telemetry and Commanding (From Ground or Stored Onboard)**
  - SRD-008 Bus Telemetry Service, SRD-009 Payload Telemetry Service, SRD-037 Telemetry Delivery, SRD-038 Telemetry Rate, SRD-039 Command Rate, SRD-031 Command Receipt, SRD-032 SOH Generation, SRD-020 Command Sequencing
- **Electrical Power Management and Control**
  - SRD-026 Battery Management, SRD-028 Power Distribution, SRD-029 Power Generation
- **Thermal Monitoring**
  - SRD-024 Heater Design, SRD-012 Thermal Environments, PLRD-004 Payload Thermal Environment
- **Attitude Control for pointing Solar Arrays to Sun and Antennas to Earth**
  - SRD-019 Stabilization Mode, SRD-021 Momentum Management, SRD-016 Safe Mode Design, SRD-044 Safe Mode Transition
- **Operational Requirements and Constraints**
  - SRD-001 Freeflyer Mission Duration, SRD-007 Electro Magnetic Environment, SRD-047 MET Clock, SRD-040 Boot Sequence, SRD-014 Governing Requirements
- **Fault Detection and Resolution**
  - SRD-043 Fault Management, SRD-017 Fault Detection

Requirements derived and BioSentinel FSW Components Allocated



BioSentinel H/W Device Availability Challenge

- **Key EDU Hardware Availability is a common FSW problem**
  - H/W Arrives Late in Development Cycle for various reasons:
    - **Long Lead Time of Components**
    - **Design/Requirement iterations**
    - **Budget/Procurement issues**
  - Limited H/W access after arrival due to:
    - **Long duration environmental tests**
    - **Demand for H/W by multiple stakeholders (testers, developers, integrators)**
- **BioSentinel Key Flight Hardware Examples**
  - Command and Data Handling (C&DH) Hardware
    - **LEON3 Single Board Computer (SBC)**
    - **Interface eXtension Card (IXC)**
  - Iris V2.1 Transponder
- **FSW in critical path**
  - Flight Software cannot finish until tested on EDU/Flight Hardware
  - Waiting for key hardware pushes delivery date
    - **Not an option for BioSentinel – short time between arrival and delivery**
- **Development/Integration without key hardware necessitates:**
  - Evolving Test Approach (WSIM, PIL, PHIL, HIL)
  - Layered Software Architecture (cFE/CFS)
  - Device Virtualization and Abstraction

H/W and Processor FPGA Virtualization (on Non-Rad-Hard consumer targets)

- **Soft-Core LEON3 Processor (NEXYS4 target)**
  - Inexpensive and can run all FSW Apps; Saves the day!
  - Useful, but Hardware Interfaces not flight-like
- **Iris Transponder H/W Simulation (ML605 target)**
  - Emulates low-level SPI Interface; big risk reduction!
  - Useful, but no external tlm/cmd I/F for integrated testing
- **BioSentinel IXC (Interface eXtension Card) (A3PE-STARTER-KIT-2 target)**
  - SpaceWire signal issues ; learning experience

NASA Ames Flexible FSW Infrastructure for Evolving Development

LADEE (Lunar Atmosphere and Dust Environment Explorer) Award Winning Effort

All Mission Goals Exceeded

200% of Required Science Data Downloaded
Lunar Laser communication successfully demonstrated
Extended Mission

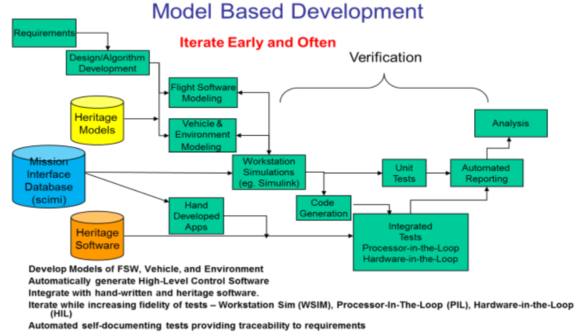
- **Survives eclipse**
- **Allows Very Low Altitude Science**

Lots more publically available information about LADEE FSW:
http://ieeexplore.ieee.org/document/7119109/
http://flightsoftware.jhuapl.edu/files/2013/talks/FSW-13-TALKS/FSW-13-GUNDY-BURLET.ppt
http://flightsoftware.jhuapl.edu/files/2014/Presentations/Day-3/Session-4/2-FSW-14-DANILO_VIAZZO.ppt

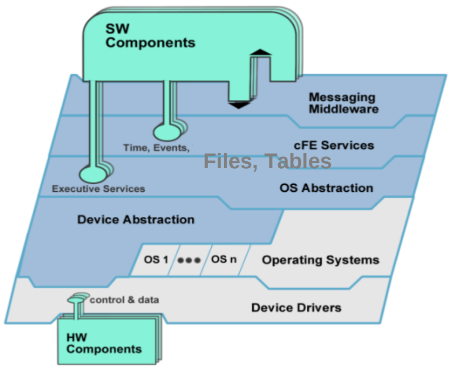
Common Avionics and Software Technologies (CAST) Lab

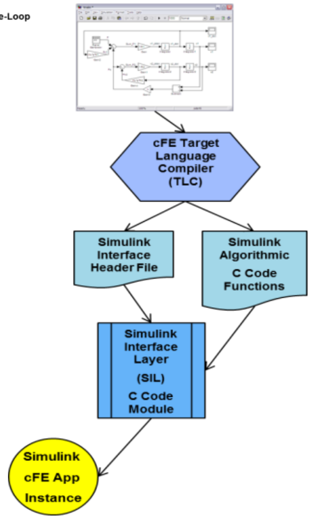
- Generalized Application of LADEE FSW Development Infrastructure
  - Re-use of software, tools, architectures, processes and **people**
  - Improvements to increase efficiency and quality
- Key Technologies
  - Model Based Development (Simulink)
    - Automatic Sim and FSW Code-Generation
    - Real-time Test-beds (PIL/HIL)
  - Loosely Coupled Component App-based real-time FSW Executive
  - Tlm/Cmd Interface Database with Code-Generation Export Capabilities
  - Continuous Build/Test Collaboration Framework
    - Bamboo / JIRA / git / Confluence

- **Model-Based Approach**
  - Auto-generation of Real-Time Code from Closed-Loop Simulink Models (Integrate early and often)
  - Integrate early and often while increasing the fidelity of closed-loop testbeds: Workstation Sim , PIL, HIL
  - Model and Evolving Test Platforms begin with initial analysis and live through Requirements, Implementation and Operational phases of mission (End-to-End)

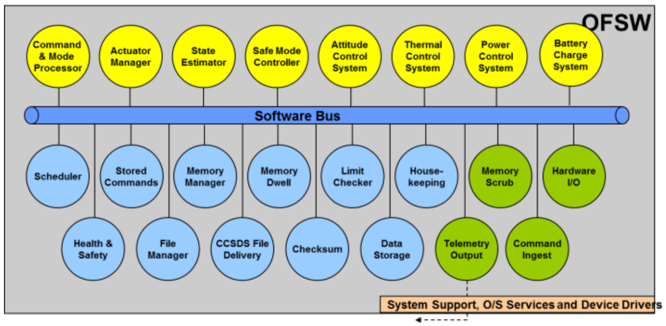


- **Core Flight Software System (CFS) developed by Goddard**
  - Layered, App-Based Architecture facilitates scalability, portability and FSW re-use
  - Core Flight System (cFS) is flight-qualified (Class B and A) and has a legacy of 11 successful NASA missions, in use at 6 NASA centers
  - Operating System Abstraction Layer (OSAL) eases porting
  - Pub/Sub Software Bus for Loosely Coupled Architecture





- **Simulink Interface Layer (SIL)**
  - Core Flight Executive (cFE) Target Language Compiler (TLC) generates cFE Pub/Sub and Table Interfaces from Simulink Models to create unique cFE App Instances
- **Device Abstraction/Virtualization**
  - Hardware I/O (HWIO) Apps provide interface between Devices and Software Bus. Leverages Posix Device I/O



Recurring CAST Themes

- “Don’t reinvent the wheel … ”
  - “A more elegant, optimized implementation” is not a driving reason
- “… but watch for square peg in round hole”
  - Don’t expect full re-use of code not written for re-use
  - Sometimes a new “wheel” is required
- “Don’t repeat yourself”
  - Eliminate redundancy; i.e. Eliminate redundancy :-)
  - Auto-generate from common data/model source and Automate repetitive tasks
- “Integrate Early and Often”
  - Evolving integrated test-beds
- “Test as you fly”
  - Use Simulation Scenarios with to make FSW think it is flying
  - Use Flight Ground System Scripts for unit and integration tests
- “Keep it Simple for the User”
  - Higher Abstraction Layers Simple; Specific Complexity (Polymorphism) in Lower Layers.

CAST Simulation Testbed Infrastructure with Evolving Levels of Fidelity

- **WSIM – Workstation Simulation**
  - **Closed-Loop Simulink Model**
  - **Non-realtime ; usually faster-than-realtime**
- **PIL – Processor in the Loop (realtime)**
  - **Flight-like interfaces (per ICDs) between Sim and FSW**
  - **Transport Mechanisms may be emulated with a run-time re-configuration (For example: ethernet drivers replace SPI driver)**
- **HIL – Hardware in the Loop (realtime)**
  - **Flight-like interfaces and Transport Mechanisms between Sim and FSW**
- **PHIL – Mix'n'match PIL and HIL**
  - **Some Transport Mechanism are emulated, others are flight-like**
  - **Example: All devices use flight SpaceWire interface, but the tlm/cmd interface is ethernet.**

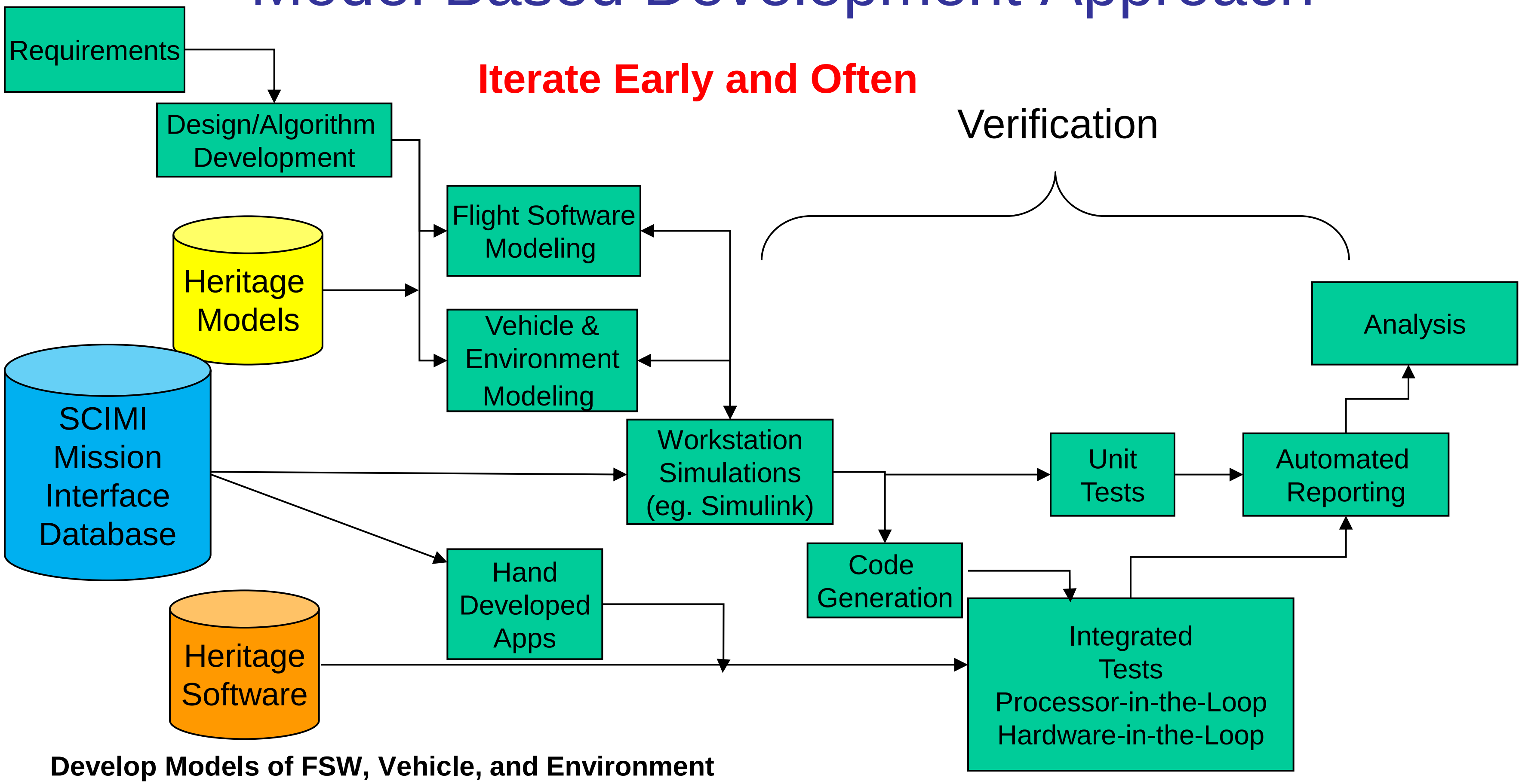
Earlier LADEE FSW Testbed presentations have more details:

- http://flightsoftware.jhuapl.edu/files/2011/FSW11_Forman.pdf
- http://flightsoftware.jhuapl.edu/files/FSW09_Gundy-Burlet.pdf

System Configuration Information & Mission Interfaces (SCIMI) Database (AKA: Cmd & Tlm Data Dictionary)

- **Relational Database Schema (i.e. metadata)**
  - Non-mission specific tables and relations
- **Mission Specific Content (independent of schema):**
  - Contains Mission information such as tlm/cmd message definitions (internal and external)
- **Built with Django Framework (open-source)**
  - Implemented with python classes
- **Non-mission specific python extraction tools auto-generate products from single definitive source**
  - C header files, matlab / simulink interfaces, Ground Software Config Files, html documentation, etc.

Model Based Development Approach



Develop Models of FSW, Vehicle, and Environment

Automatically generate High-Level Control Software

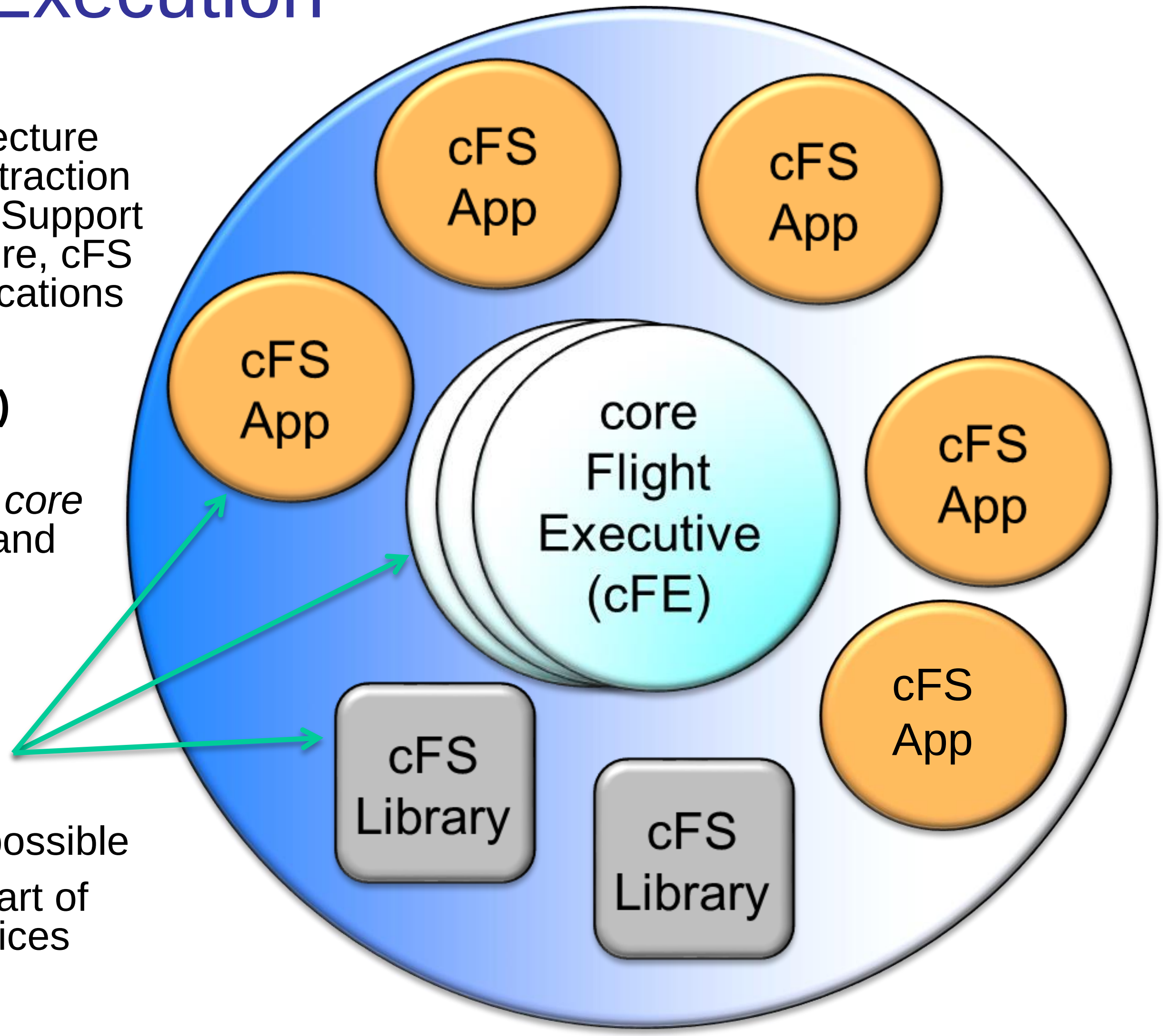
Integrate with hand-written and heritage software. Interface code generated from the SCiMI

Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)

Automated self-documenting tests providing traceability to requirements

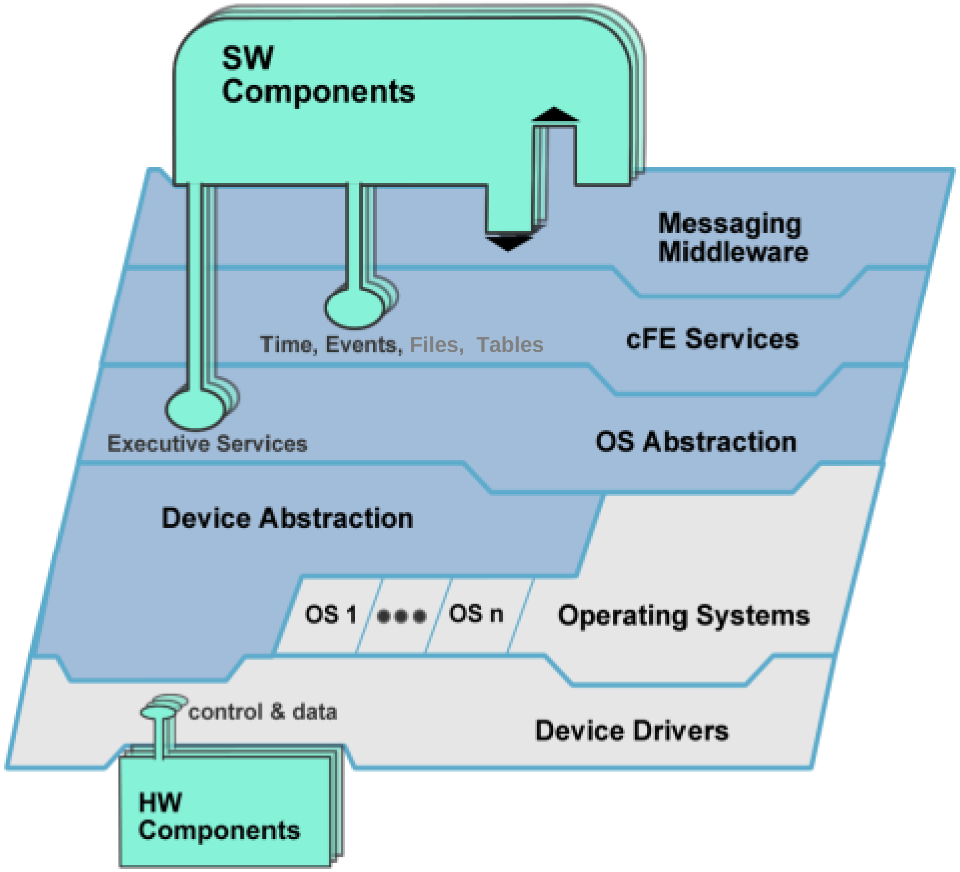
cFS Elements for Real-Time FSW and/or SIM Execution

- **core Flight System (cFS)**
  - A Flight Software Architecture consisting of an OS Abstraction Layer (OSAL), Platform Support Package (PSP), cFE Core, cFS Libraries, and cFS Applications
- **core Flight Executive (cFE)**
  - A framework of *mission independent, re-usable, core* flight software services and operating environment
- **Each element is a separate loadable file**
  - Divide and Conquer
  - Independent unit tests possible
  - In-Flight Uploaded/Restart of App built into Core Services



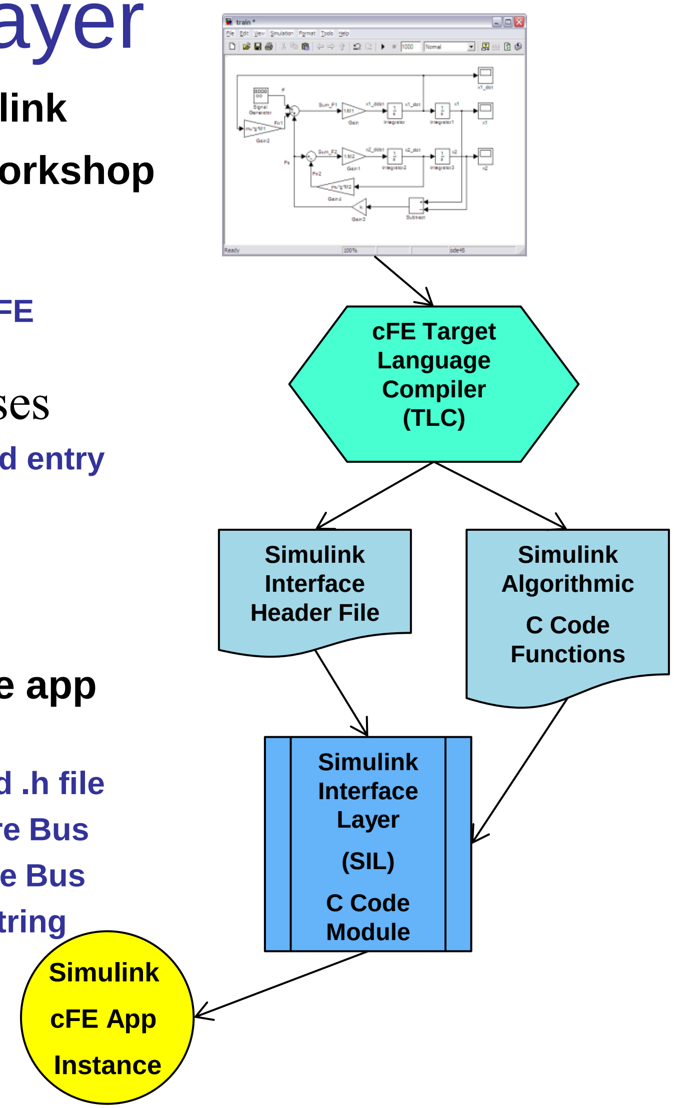
cFE / CFS Layering

- Each layer "hides" its implementation and technology details.
- Internals of a layer can be changed -- without affecting other layers' internals and components.
- Enables technology infusion and evolution.
- Doesn't dictate a product or vendor.
- Provides Middleware, OS and HW platform-independence.

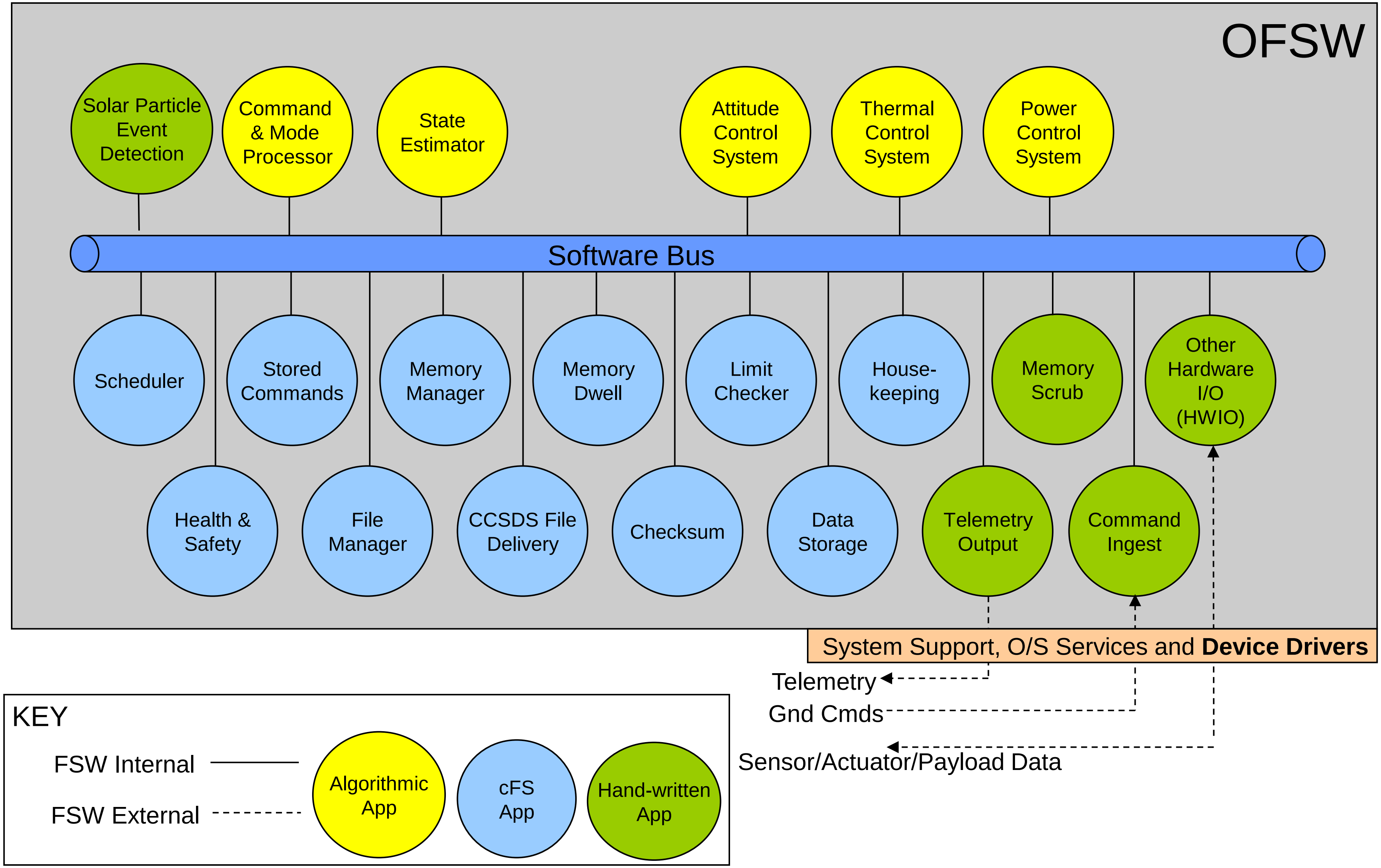


Simulink Interface Layer

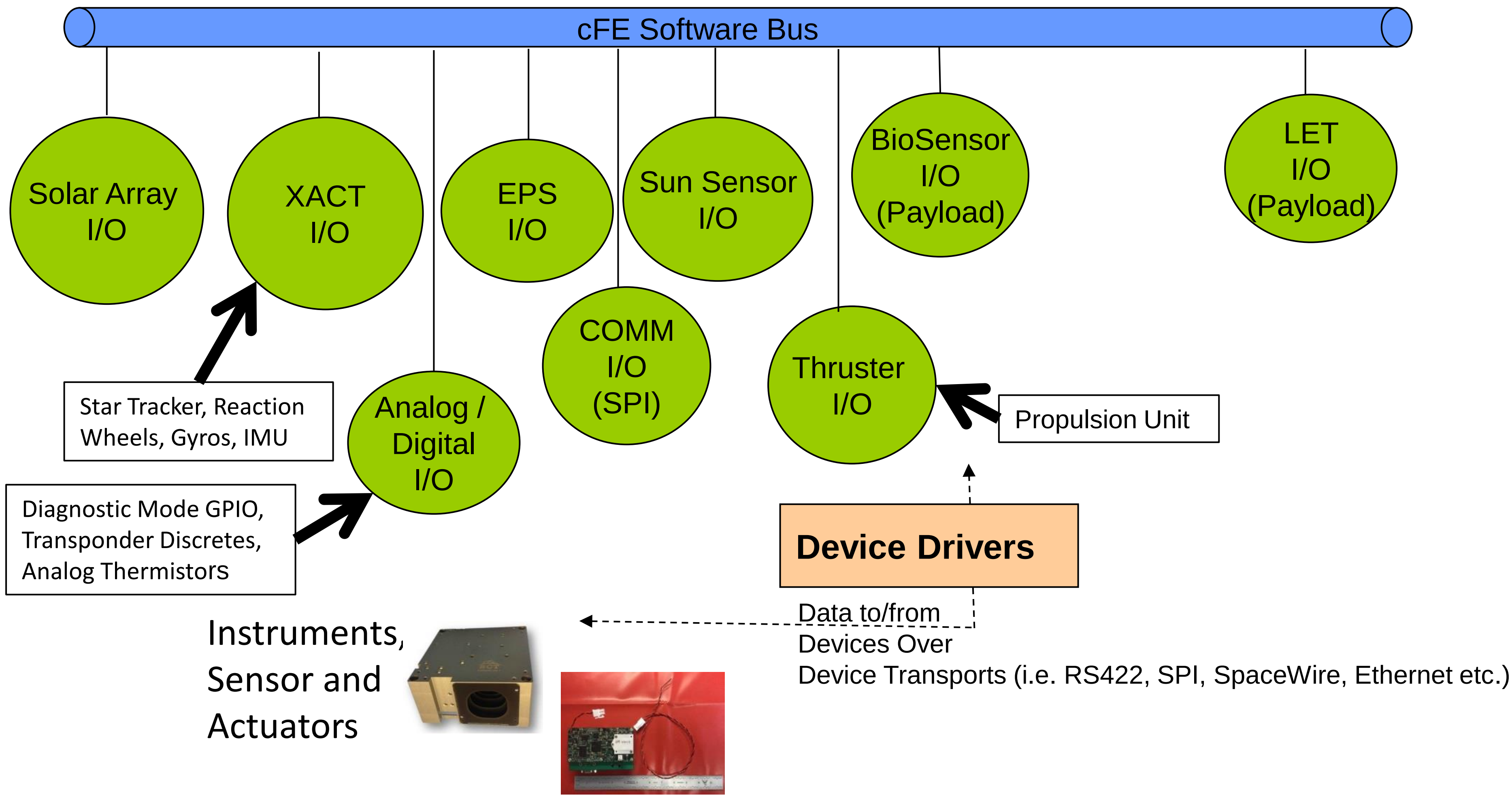
- **Higher level Flight Software Modules modeled in Simulink**
- **C-Software generated from Models using Real Time Workshop Embedded Coder**
  - cFE Template for Target Language Compiler (TLC)
    - **Transforms Specified Simulink Input/Output ports into cFE Message structures**
      - I/O ports are Simulink non-virtual buses
    - **Creates C Header file that defines message interfaces and entry points**
      - Specific Data Structures
      - Macros that identify key functions
- **Simulink Interface Layer (SIL) provides cFE compatible app functionality:**
    - **Uses message and entry point definitions from generated .h file**
    - **Input Messages – Subscribed to and recv'd from Software Bus**
    - **Output Messages – Registered and Published to Software Bus**
    - **Event Output – Custom Block with Trigger and Format String**
    - **Table Management – Mapped from tunable params**
    - **Housekeeping – General Meta-Data about App**
    - **Execution - Triggered by Specified S/W Bus Message**



BioSentinel “Loosely-Coupled” S/C FSW Architecture ; Independent Apps



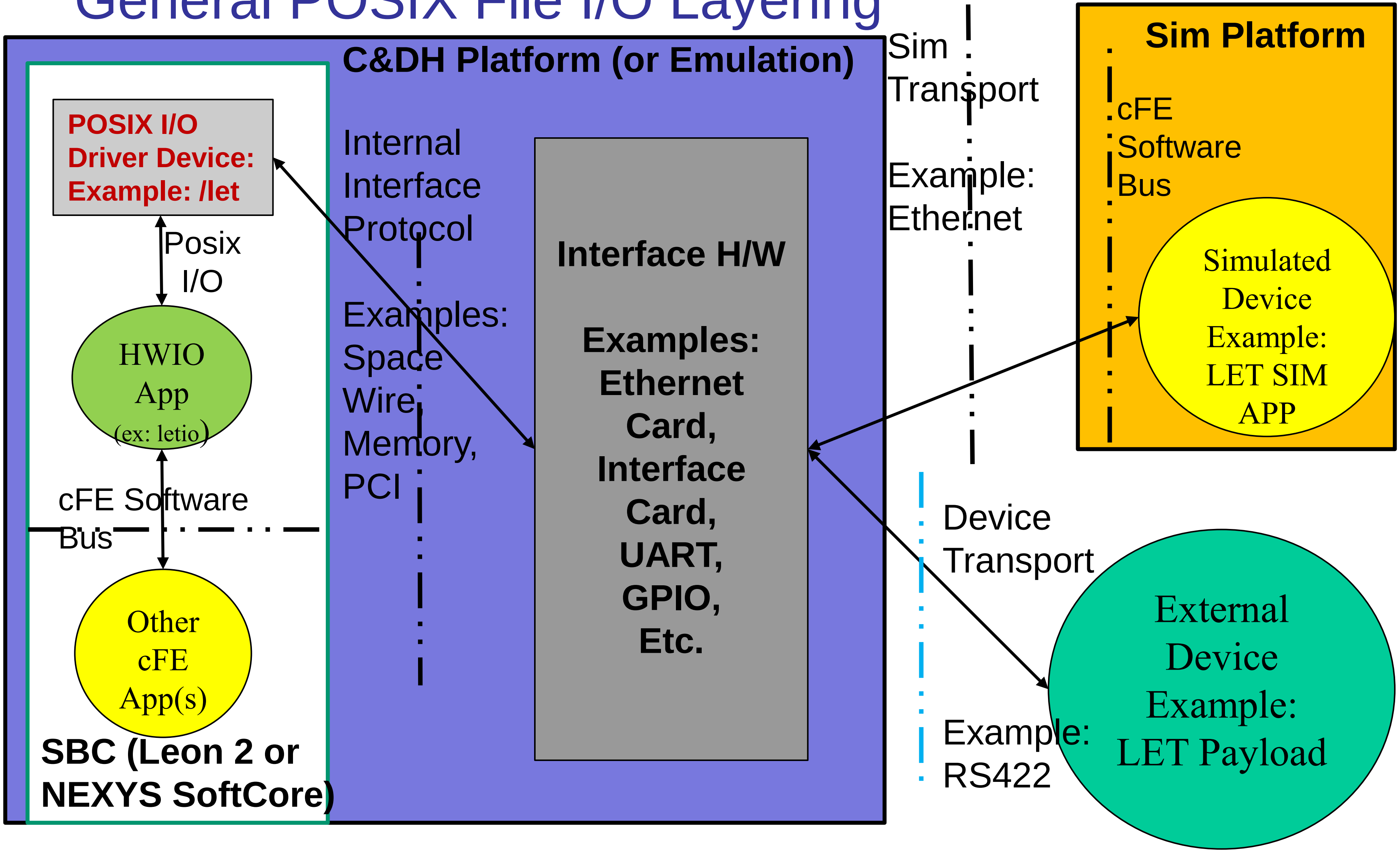
HWIO Apps in BioSentinel FSW Architecture



FSW Device Abstraction/Virtualization

- **Algorithmic Apps access data published to the cFE Software Bus by HWIO Apps**
- **HWIO Apps act as device protocol adapters**
  - Adapt device data to/from S/W Bus Msgs
- **HWIO Apps access H/W using a standard Posix File I/O Device Driver layer**
  - Open, read and write calls are the same for a simulated or real H/W device

General POSIX File I/O Layering

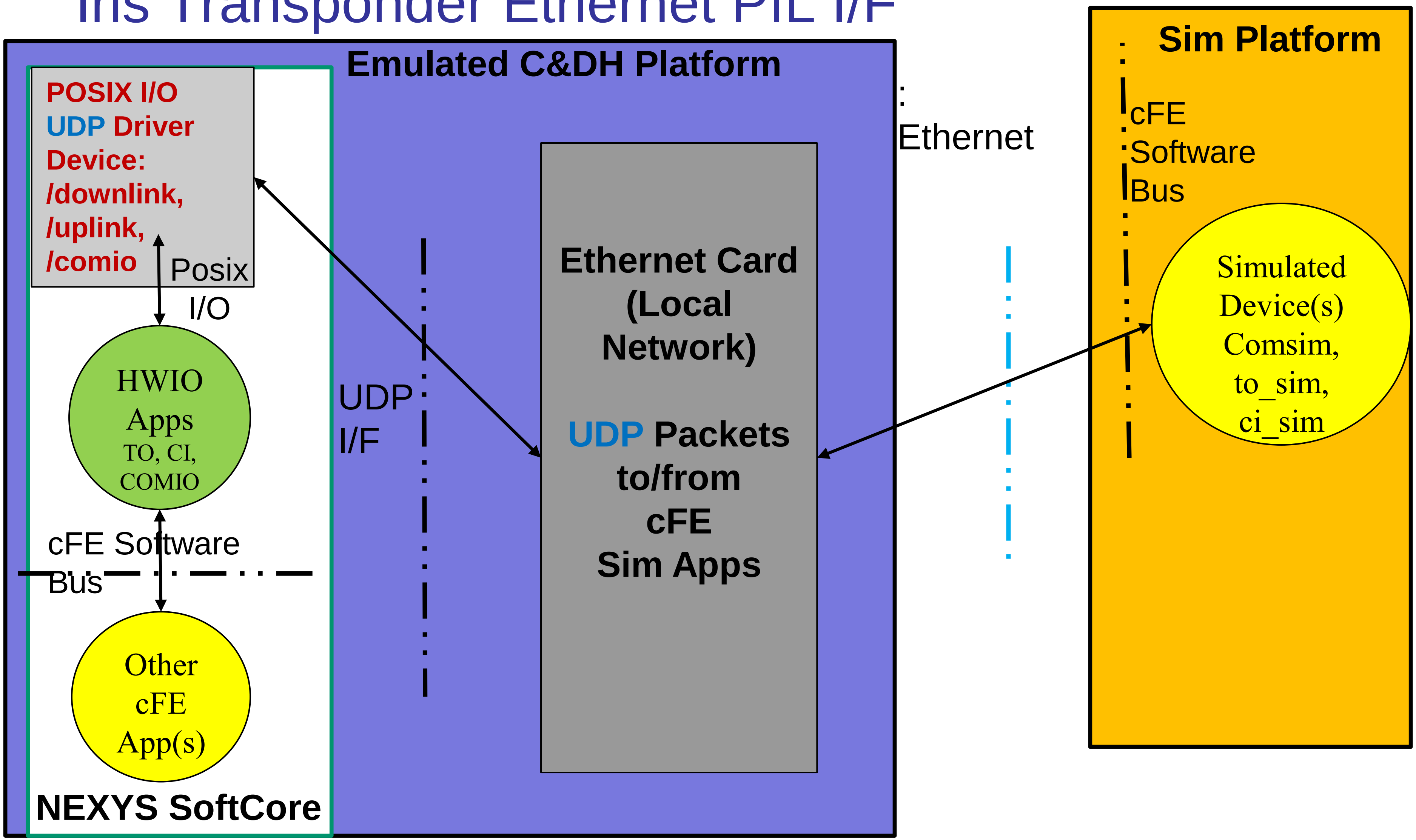


Posix I/O Layer Advantages for FSW and Sim

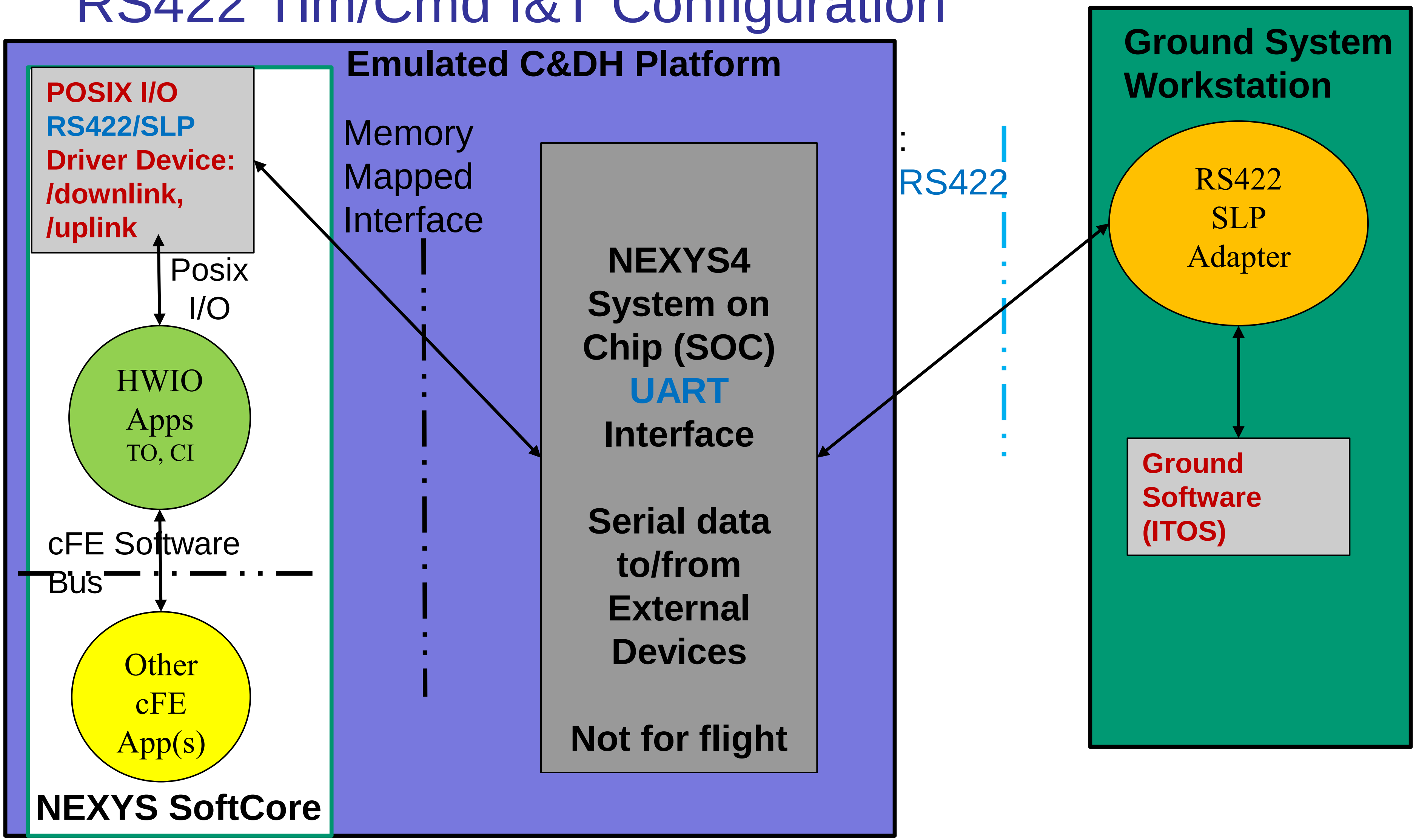
- Application Code can remain **exactly** the same whether simulated or real devices are used.
  - Only the transport driver implementation changes
  - Device Simulation is simple to implement
    - Actual files, UDP packets, Shared Memory, UARTs etc.
  - Facilitates Inexpensive Development Platforms that exercise all App code (including HWIO) in an integrated fashion.
  - Same App binaries run on all targets
- Allows Mix'n'Match evolution of devices
  - Drivers can be over-ridden (re-loaded) at runtime
- Protocol code resides in HWIO Application
  - Device Cmd/Tlm interface code tested on all targets
  - Realistic Device Virtualization (same in flight and sim envs)
- HWIO can be developed before H/W arrives.

Iris Transponder I/F FSW Evolution

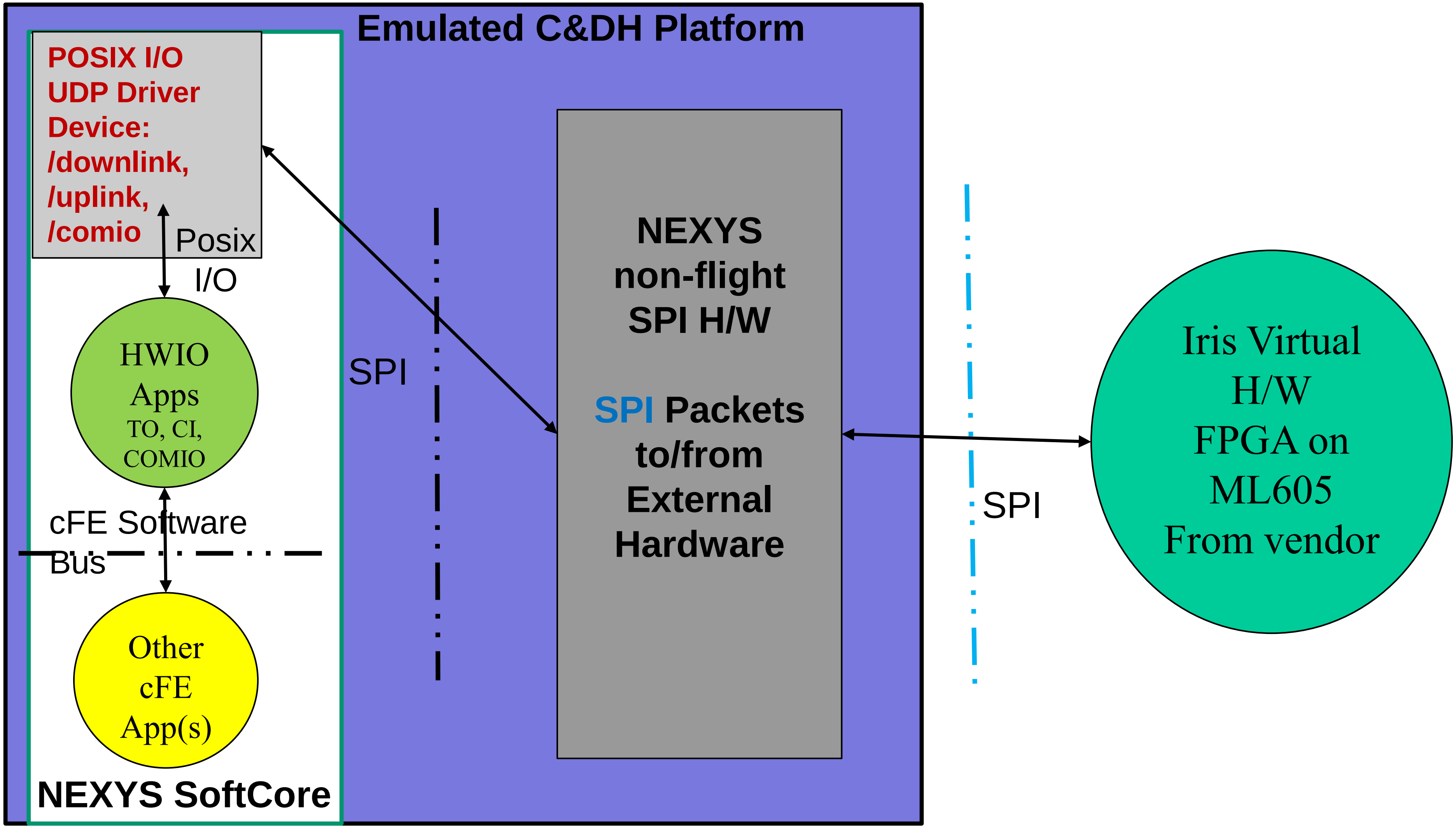
Iris Transponder Ethernet PIL I/F



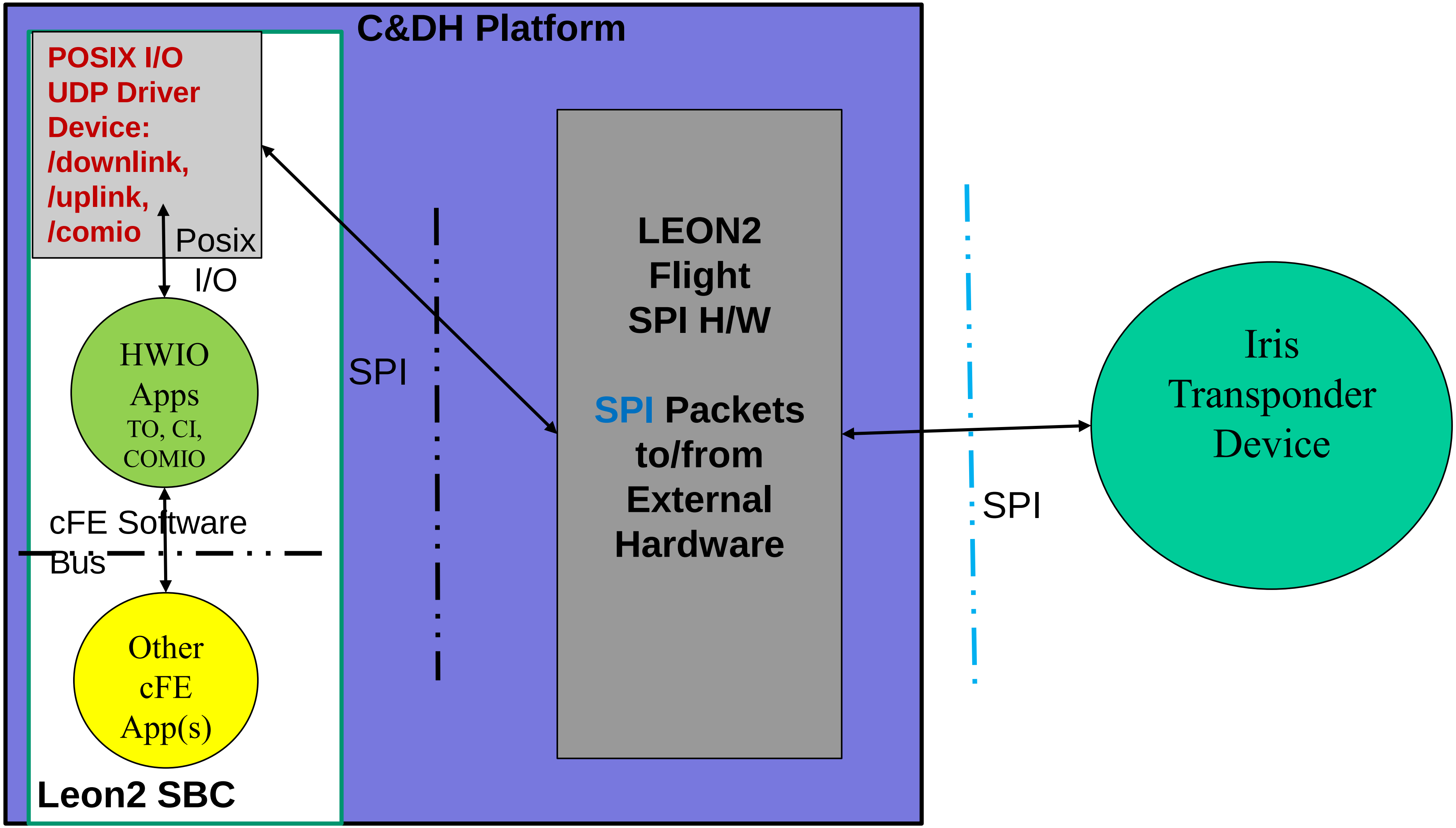
RS422 Tlm/Cmd I&T Configuration



Iris Transponder SPI I/F to Emulated Iris H/W

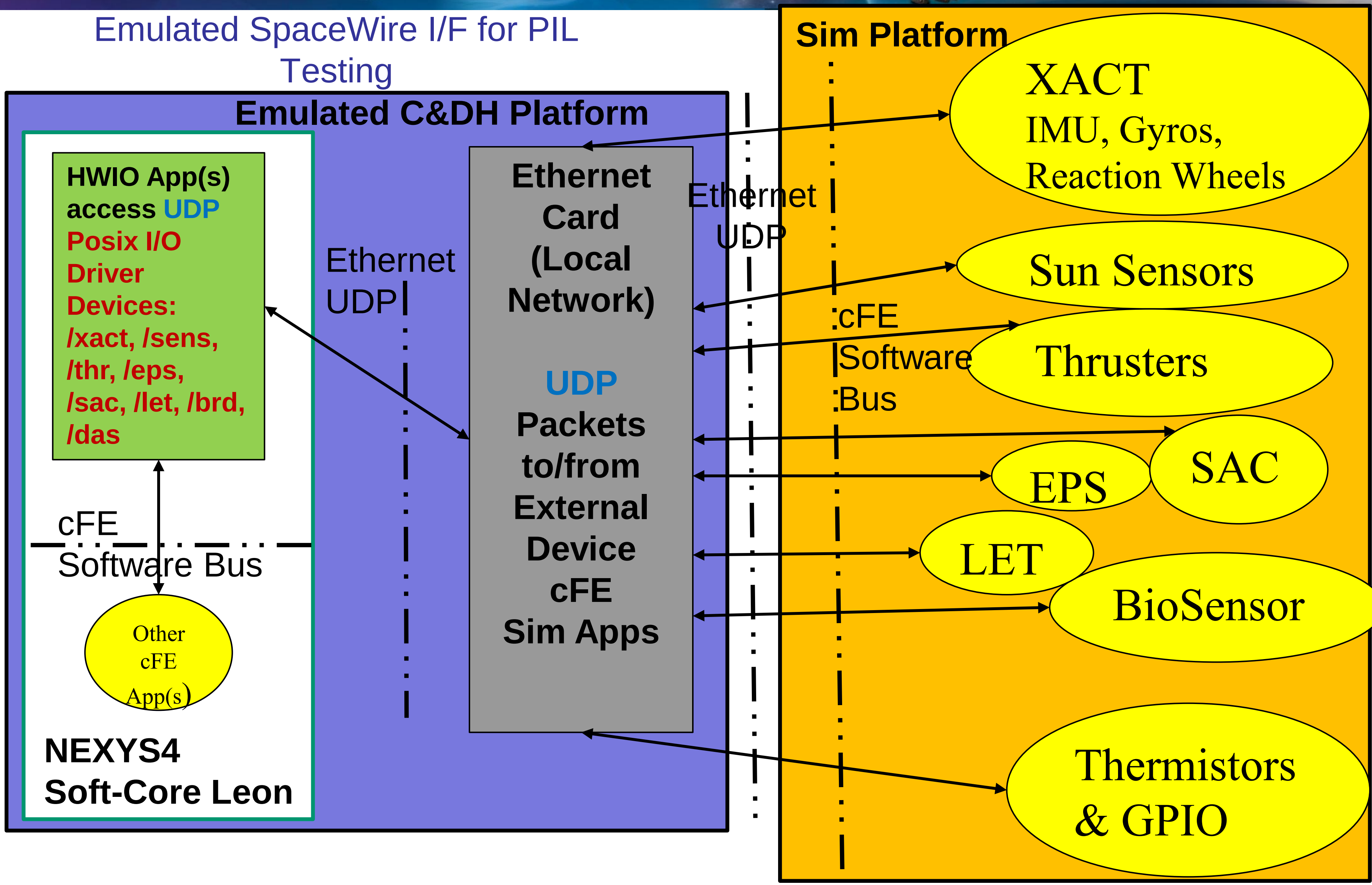


Iris Transponder Flight Interface

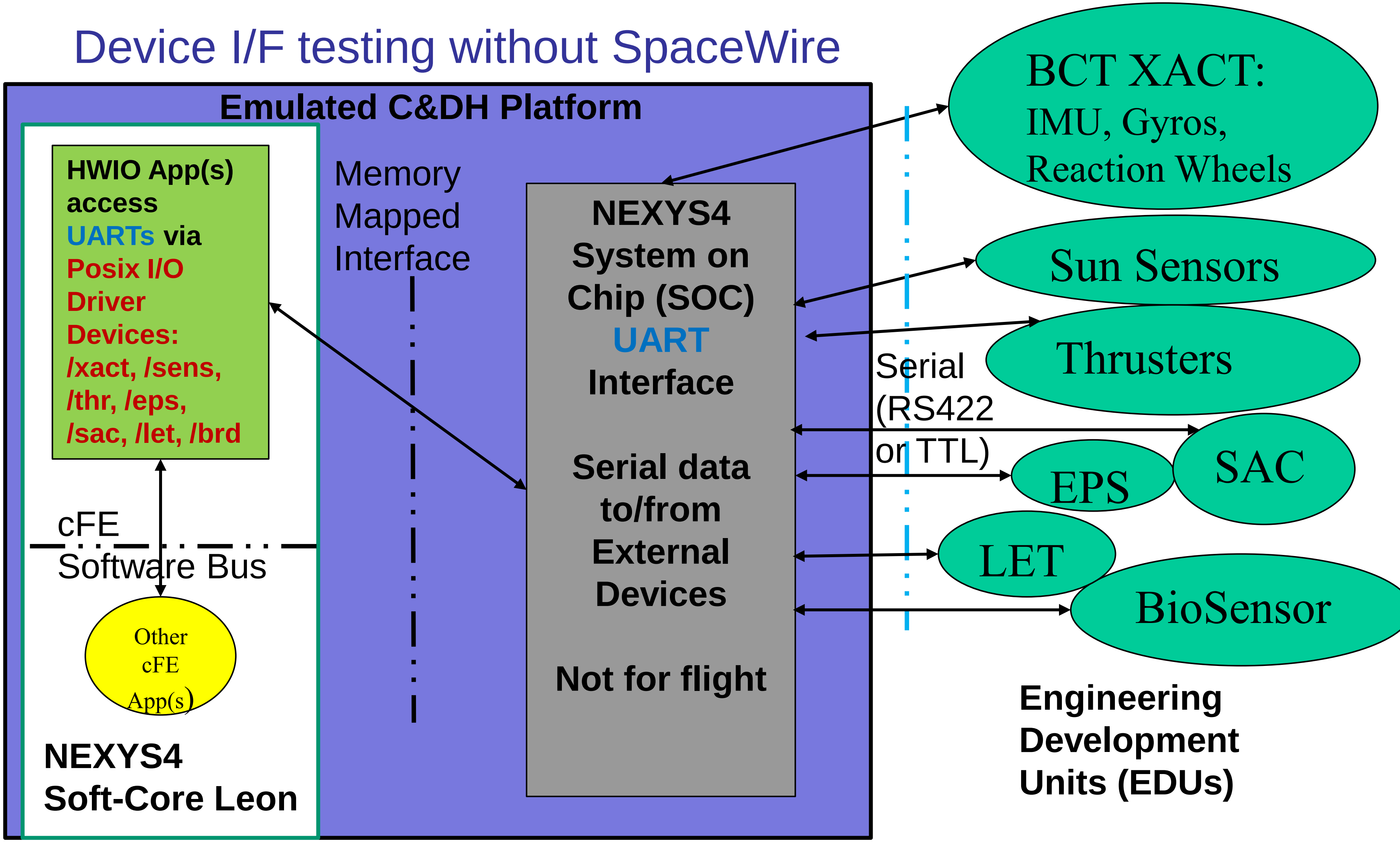


C&DH Interface eXtension Card (IXC) Spacewire-to-Serial FSW Evolution

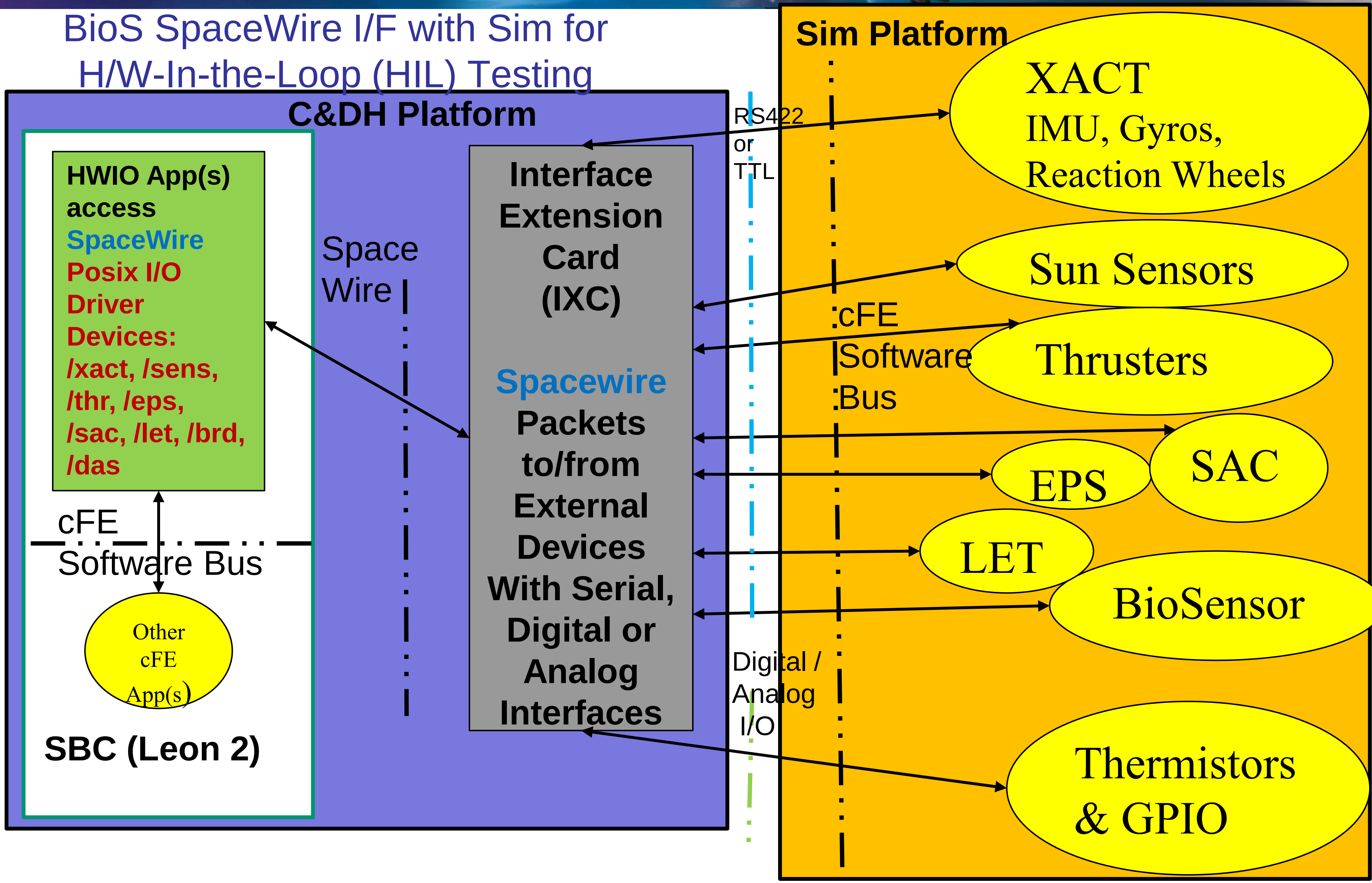
Emulated SpaceWire I/F for PIL Testing



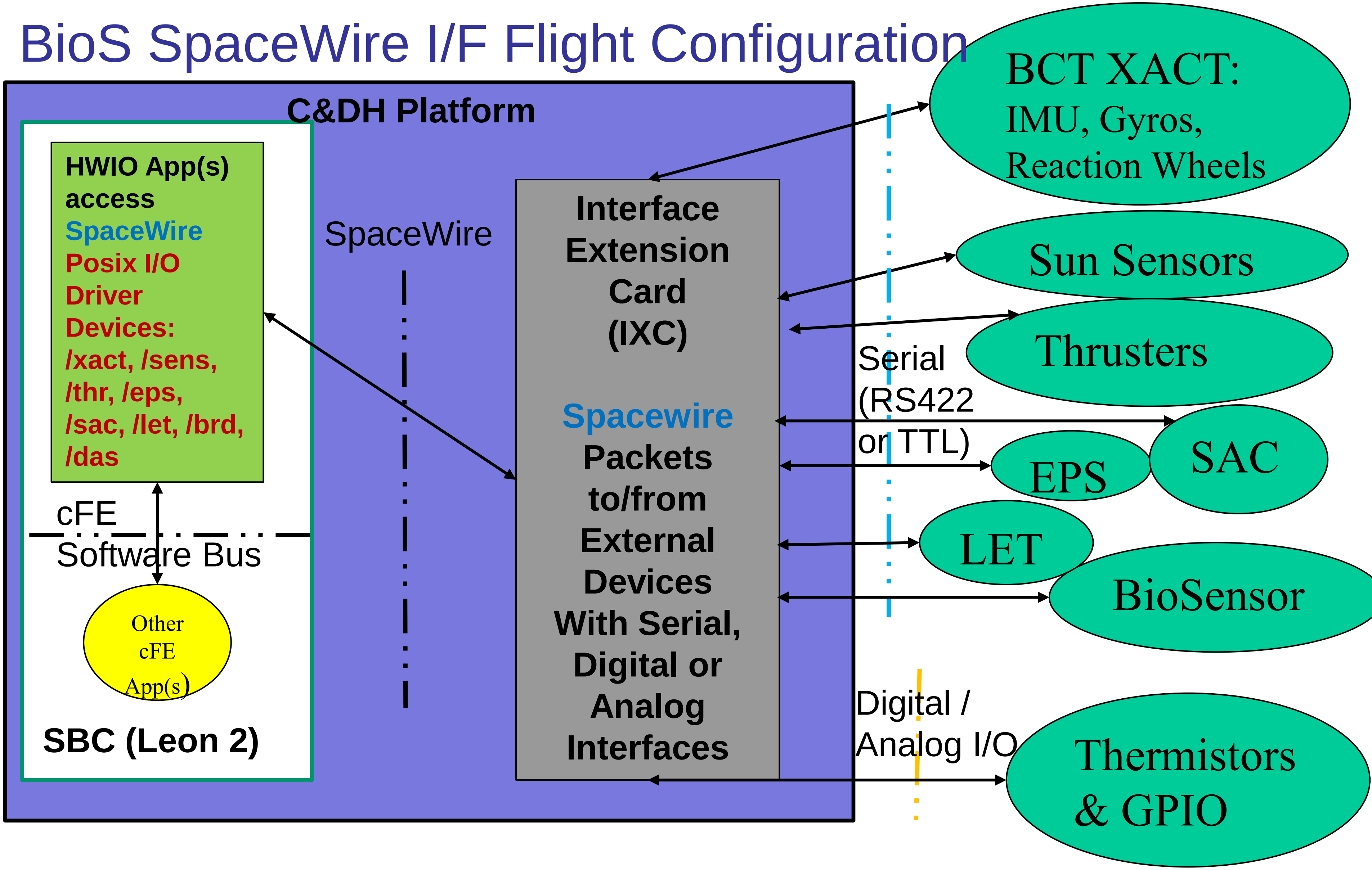
Device I/F testing without SpaceWire



BioS SpaceWire I/F with Sim for H/W-In-the-Loop (HIL) Testing



BioS SpaceWire I/F Flight Configuration



Conclusion

- **Model-Based Approach integrated with a Loosely-Coupled and layered framework enables a Flexible Infrastructure**
- **Flexible Infrastructure with Evolving Layers and Test-Beds helps meet Hardware Availability Challenges**

Questions?

Douglas.a.Forman@nasa.gov

Back-Ups

Flight Software Guiding Principles

- COTS/GOTS elements used wherever possible
- Many flight software elements are modeled in MATLAB/Simulink and then auto-coded into software applications
  - Utilities → Computer Software Components (CSCs) → Computer Software Units (CSUs)
- Workstation simulation (WSIM) → processor-in-the-loop (PIL) testing → hardware-in-the-loop (HIL) testing
- Make use of the open source core Flight System (cFS) and core Flight Executive (cFE) open source architecture

Flight Software Nomenclature

- Workstation Simulation (WSIM)
    - FSW and Plant Simulink Models
- Processor In the Loop (PIL)
    - Emulated Hardware Device Interfaces
- Hardware In the Loop (HIL)
    - Flight-like Hardware Device Interfaces
- Glue Code
    - A code base and set of scripts to automate the creation of cFS applications from Model Based Designed missions
- SCIMI (System Configuration Information and Mission Interfaces)
    - Django/Python Relational Database Framework
    - Flight/Test/Ground Files auto-generated

Recommended External Device Data Interface (CCSDS Space Packets)

Standardized Serial Data Interface Protocol

Developed in conjunction with Mission Operations Lead

Distributed to External Vendors

CCSDS Space Packets

- **With cFE compatible secondary cmd/tlm headers**

For new software on External Devices

Wrapper approach for Legacy Devices (i.e. LET)

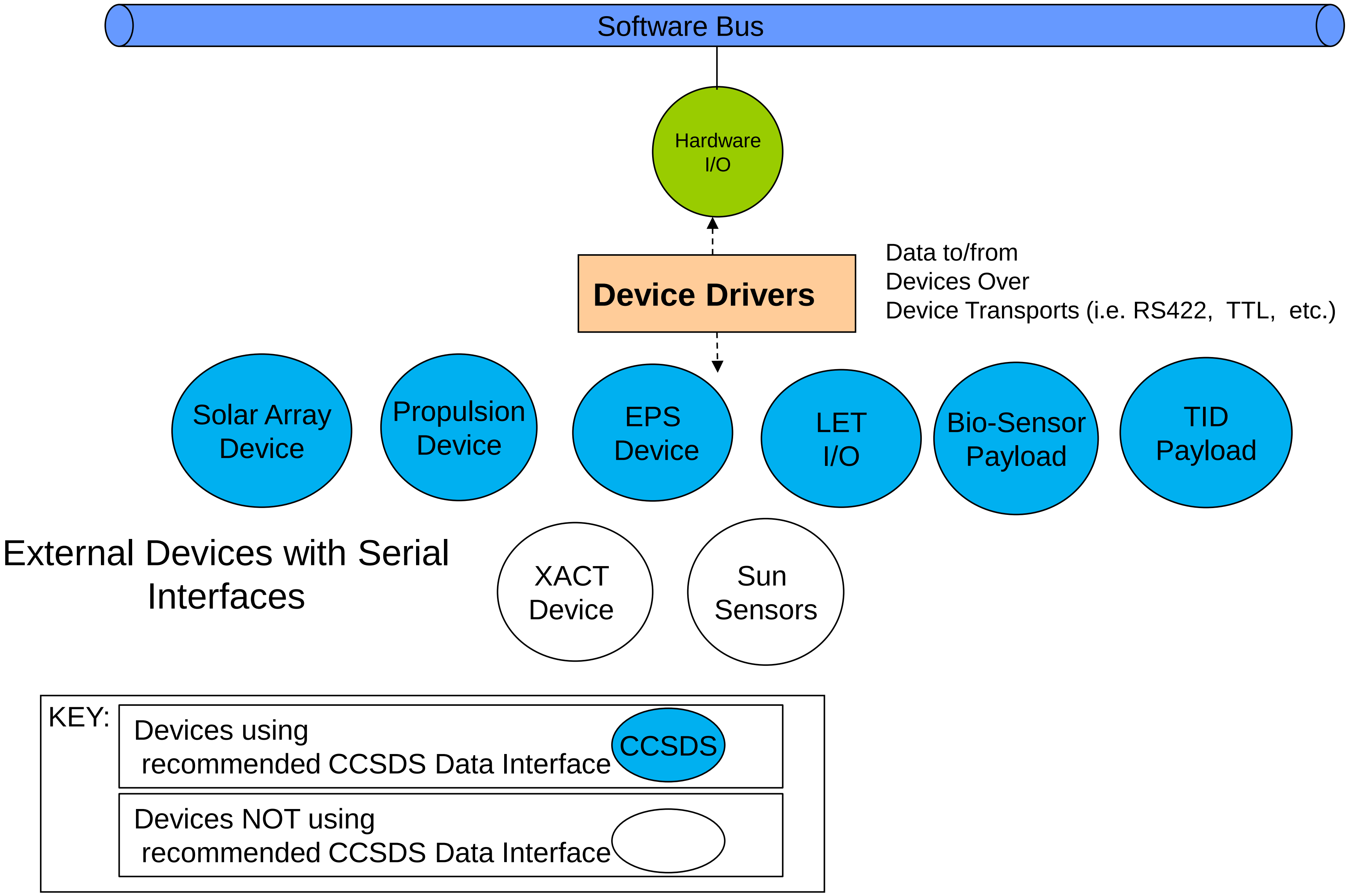
Advantages:

Device data packets can go directly to/from cFE S/W Bus

Ground System can interface directly with device over RS422

Several Device Vendors have committed:

BioSensor, LET, EPS controller, TID, Prop System, Solar Array Controller


Software Bus
Hardware I/O
Device Drivers
Data to/from Devices Over Device Transports (i.e. RS422, TTL, etc.)
Solar Array Device
Propulsion Device
EPS Device
LET I/O
Bio-Sensor Payload
TID Payload
External Devices with Serial Interfaces
XACT Device
Sun Sensors
KEY:
Devices using recommended CCSDS Data Interface
CCSDS
Devices NOT using recommended CCSDS Data Interface

National Aeronautics and Space Administration

Ames

Discovery Innovations Solutions

WSIM **Workstation Simulations**	**Simulink on Windows, Mac, or Linux computers**	**•Models of GN&C, Prop, Power, & Thermal** **•Used by FSW to generate and test algorithms.** **•Provided to MOS for limited functionality maneuver simulations.**
PIL **Processor-in-the-Loop**	**PPC750 Processor(s) in Standalone chassis** **or** **NEXYS4 Soft-core LEON C&DH with UEI Simulation**	**•Includes all flight software functionality. Runs on 1 or 2 processors.** **•Ethernet or Shared Memory Interfaces between FSW and Simulation** **•Multiple copies maintained by FSW as inexpensive system for real time software & fault management development.** **•Faster than real time (depending on selected fidelity of models and processor speed.**
HIL **Hardware-in-the-Loop**	**Avionics EDU with simulated vehicle hardware (UEI Racktangle Simulation Platform)**	**•Highest fidelity simulators includes hardware interfaces.** **•Run in real time.** **•Travelling Road Show used to test payload interfaces early in development cycle** **•Authoritative environment for verification of FSW requirements**

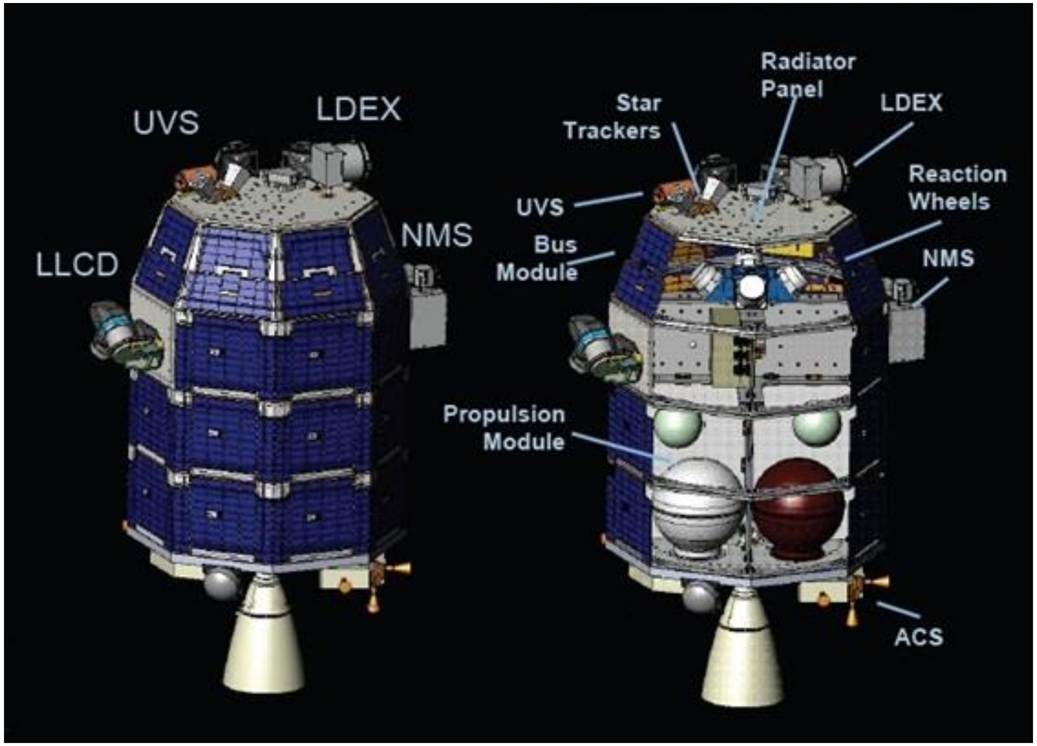
Modular, Common Avionics and Software

Integrated with a range of processors

- Trade study to make it easier for spacecraft to identify acceptable processors with necessary performance and budget constraints.
- Initial Processors targeted:
  - Beagle Bone Black
  - Zynq
  - PowerPC 750
  - LEON

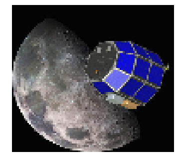
- Provide a set of Software Engineering Practices and Documentation to quick start spacecraft software development effort
  - Git
  - Confluence/JIRA
  - Bamboo Continuous Test
  - NPR 7150.2 required plans
  - Extensive Test Suite
  - Peer Review/Formal Inspection

The Lunar Atmospheric Dust Environment Explorer (LADEE) Spacecraft



A small explorer class mission with the objective of investigating the Moon's tenuous atmosphere, and to study the composition/distribution of dust in that atmosphere.

BioSentinel Flight Software

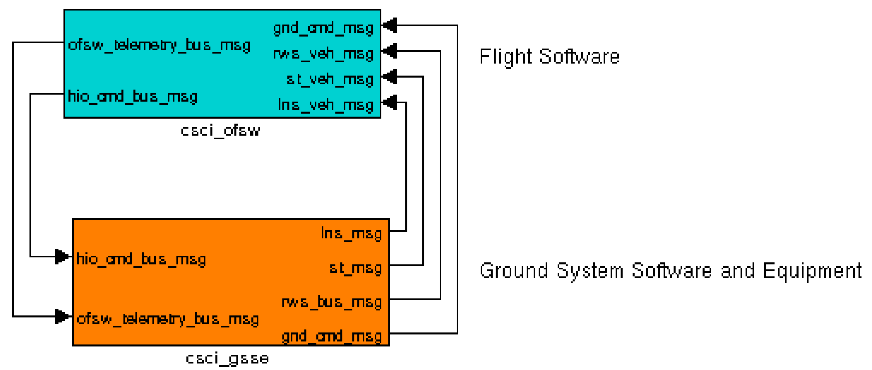

LADEE

Scope

-Onboard Flight Software (Class B)
-Support Software and Simulators (Class C)
-Integration of FSW with avionics

Guiding Documents

-NPR7150.2 Software Engineering Requirements
  -CMMI Level 2 or Equivalent
-NASA-STD-8739.8 NASA Software Assurance Standard



Development Approach

Model Based Development Paradigm (prototyped process using a “Hover Test Vehicle”)

5 Incremental Software Builds, 2 Major Releases before launch

- **Final Release during mission.**

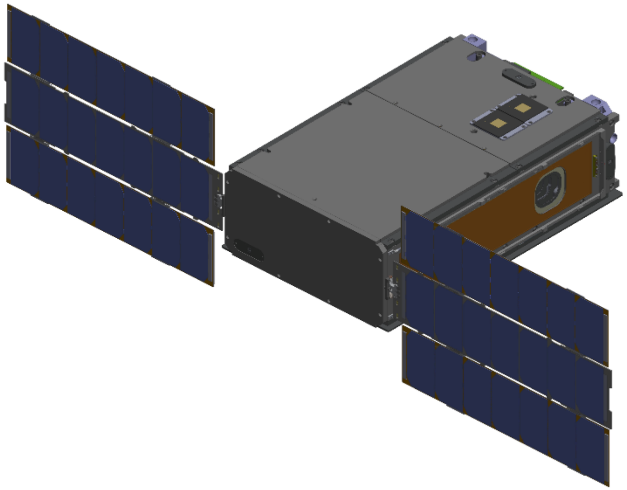
Leverage Heritage Software

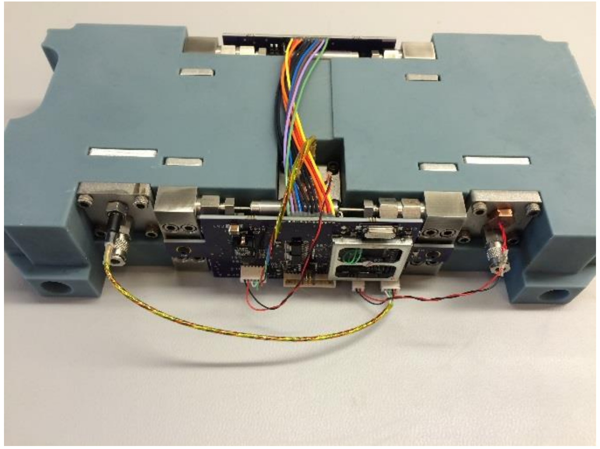
GSFC OSAL, cFE, cFS, ITOS

, VxWorks

Mathworks Matlab/Simulink & associated toolboxes

BioSentinel: A Deep Space CubeSat

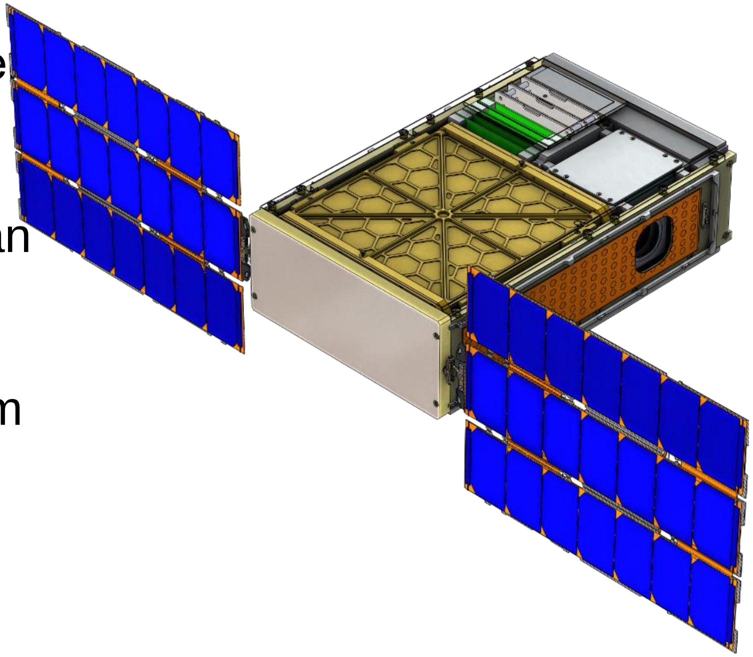




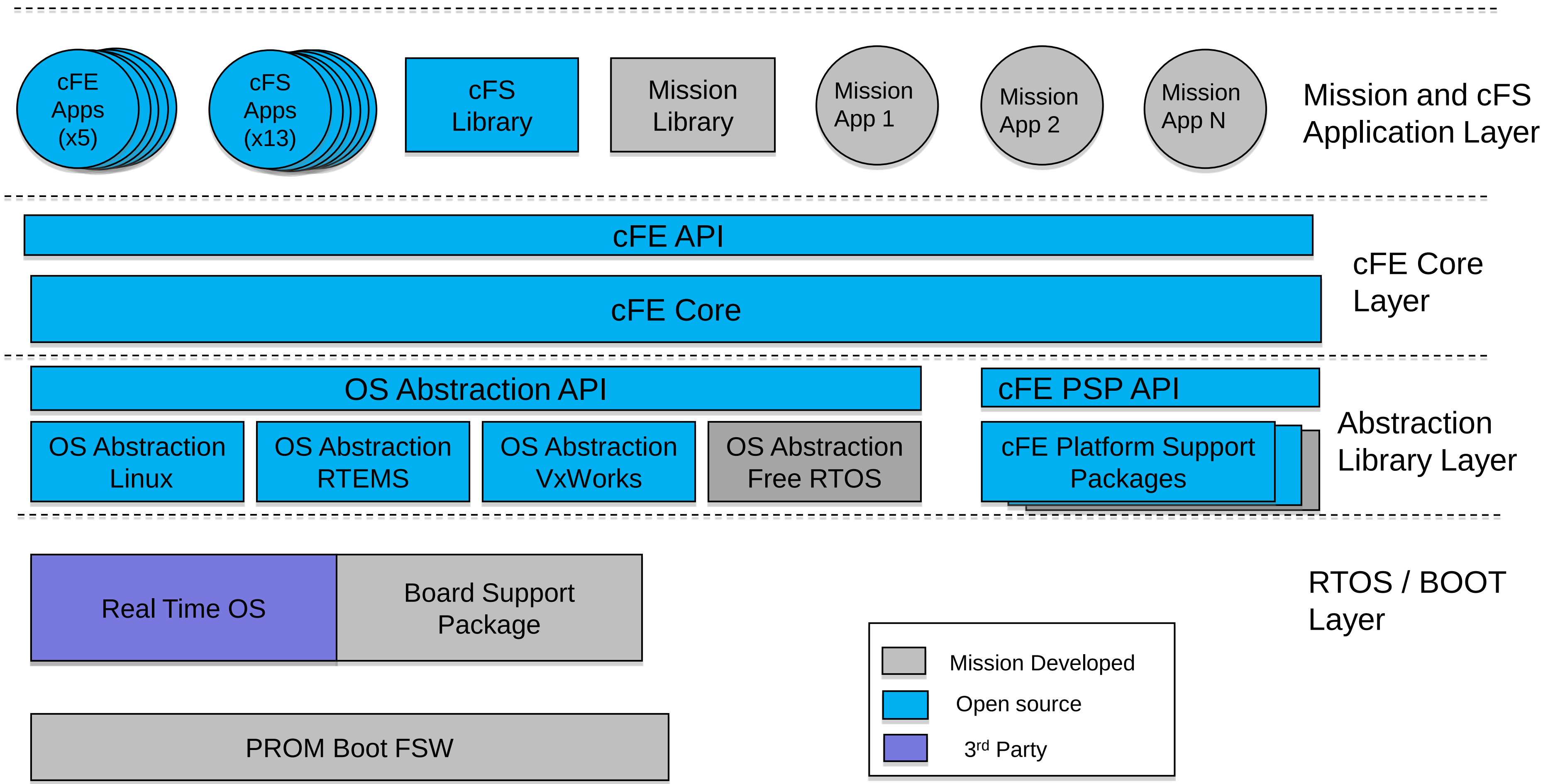
- A “CubeSat” class spacecraft with stowed outer dimensions of approximately 20cm x 30cm x 10 cm
- Science payload will study the effects of deep space radiation on cells that are similar to those of humans
- An important pathfinder to enable future human missions to the Moon, asteroids, or Mars

Small Spacecraft, Big Software

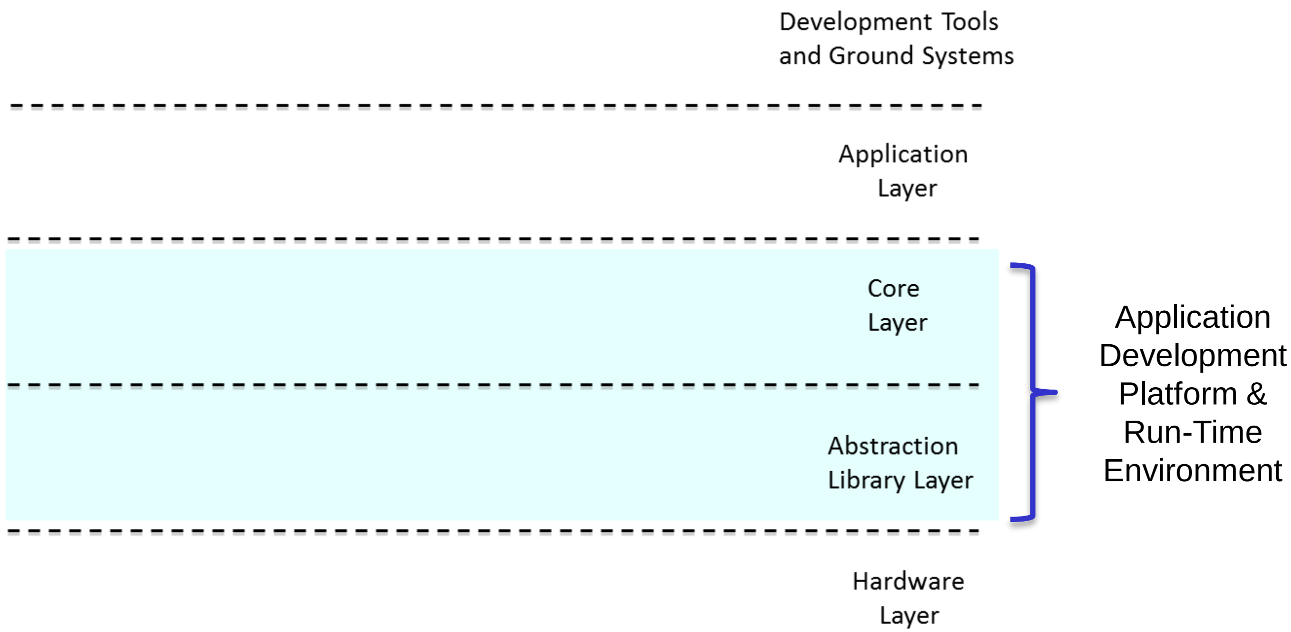
- Science payload is controlled using a TI MSP430 microcontroller
  - “Sidecar” science payload also driven with an MSP430
- Electrical power system controlled with an MSP430
- Attitude determination and control system has its own FPGA and LEON Processor
- Radio has its own FPGA and LEON Processor
- Overall spacecraft command and data handling managed with an FPGA



cFS Software Layers and Components

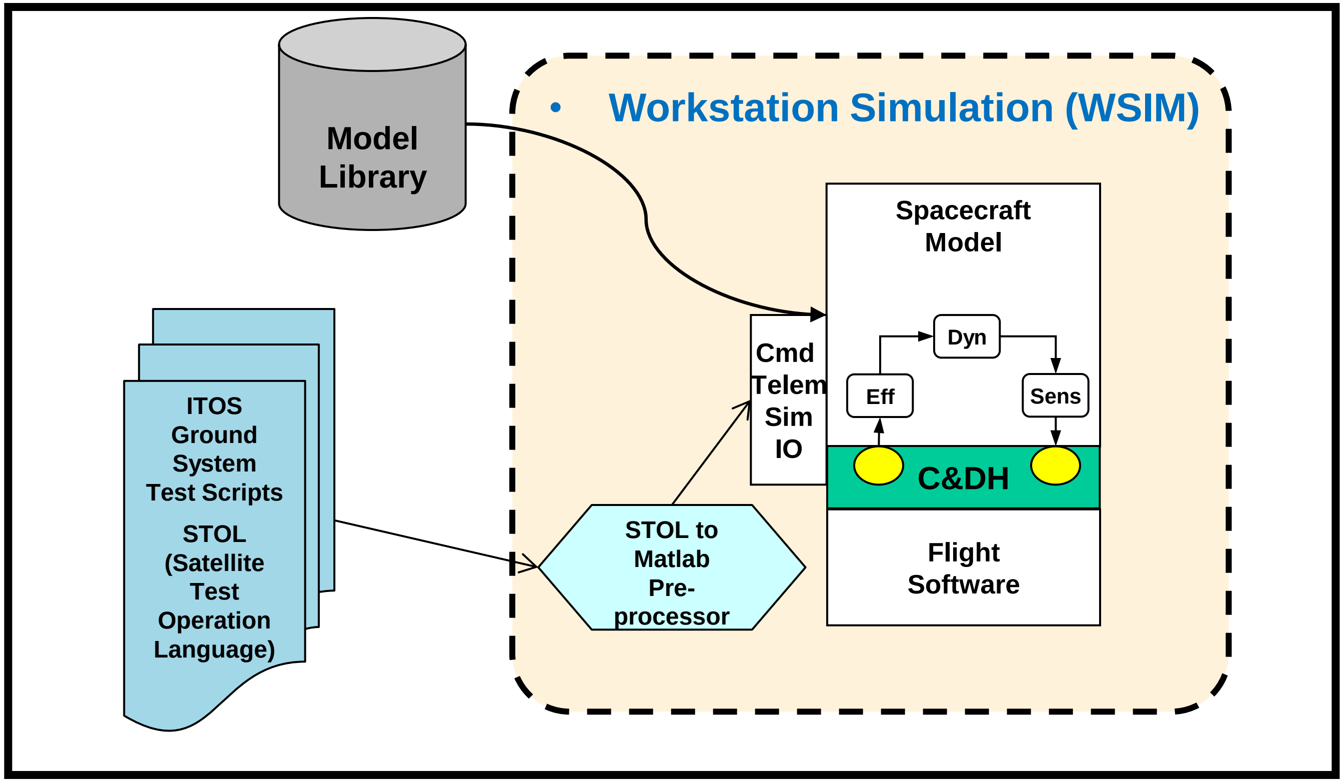


cFS Overview



cFS is a re-usable spacecraft flight software architecture and software suite that is both platform and project independent

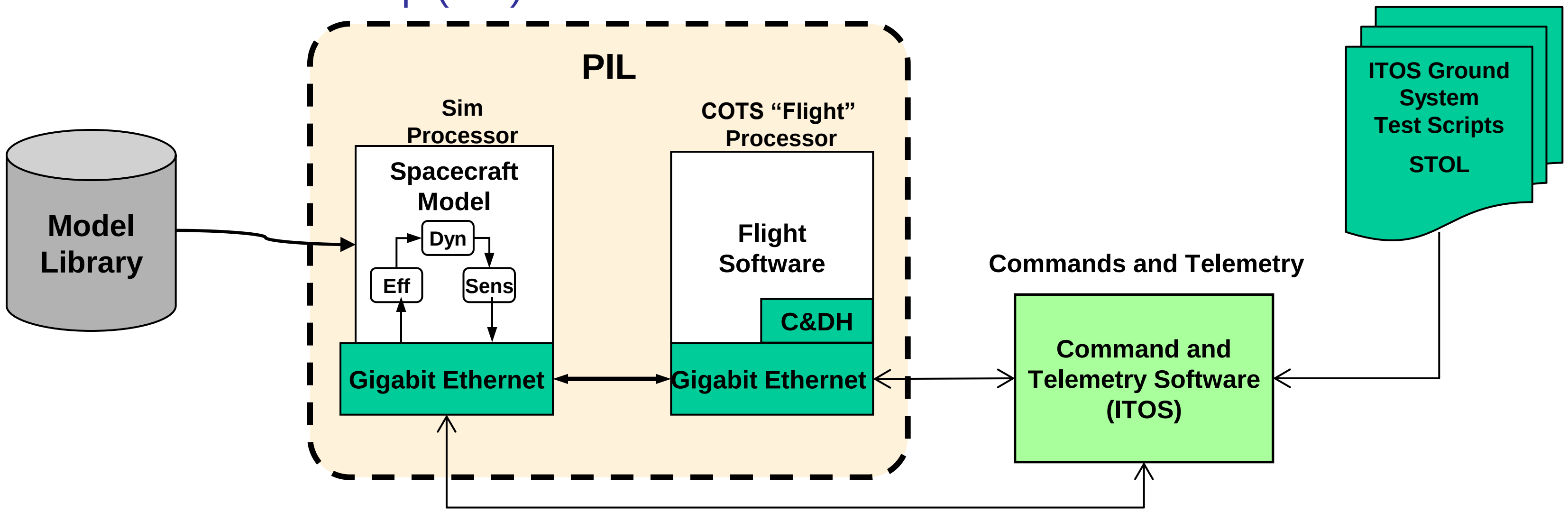
The software suite includes a common set of spacecraft applications including communications, scheduling, fault management, and many others



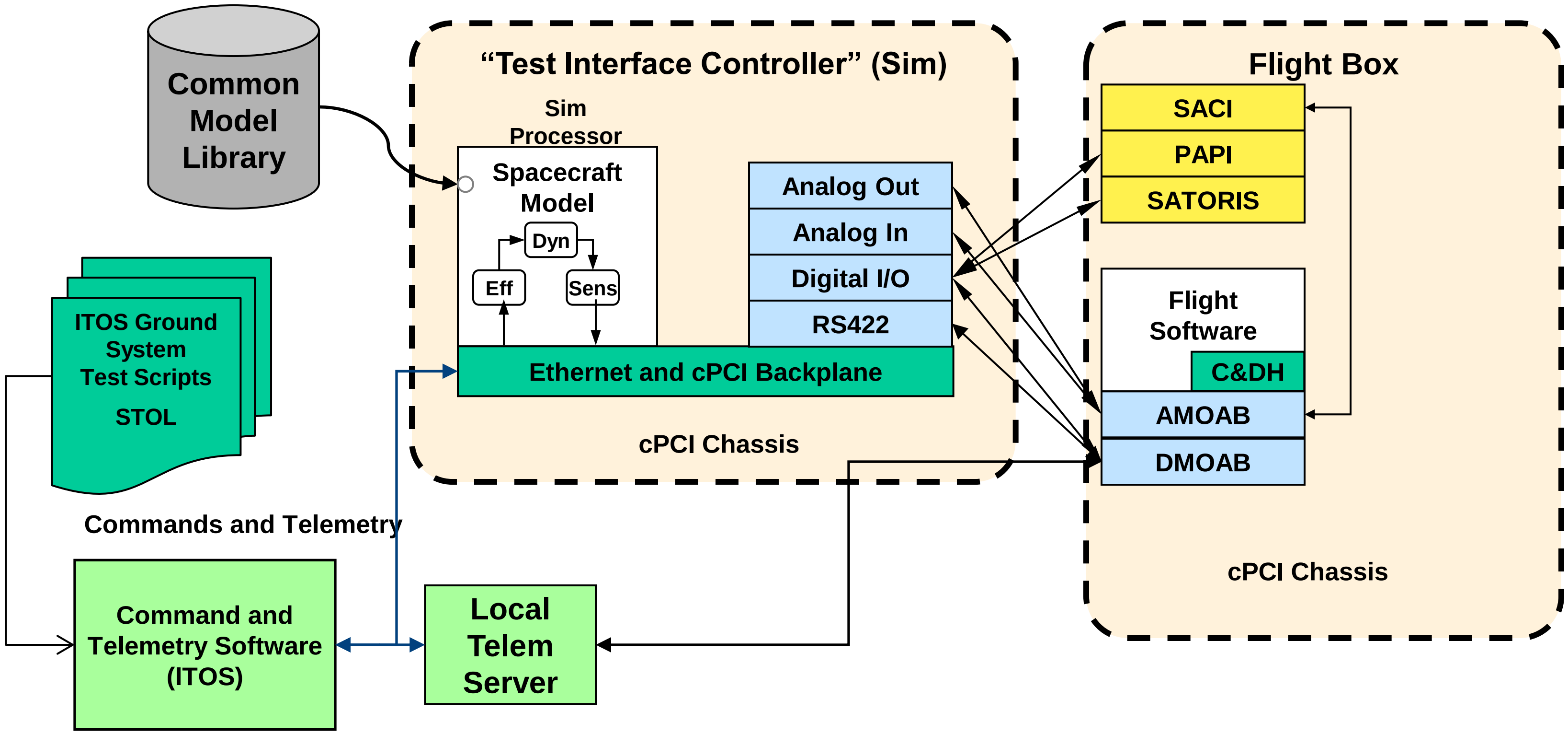
Local Workstation or Laptop

- Simulink Only (No Autocode)
- Developed early, maintained throughout development
- Algorithm Development ; Requirements Analysis
- ITOS Ground System Scripts ingested by Simulation
- cFE Services Modeled within Simulink
  - cFE Messages -> Non-Virtual Busses,
  - cFE Tables -> Tunable Parameters
  - cFE Scheduler -> Simulink Scheduler and Rate-Transition Blocks

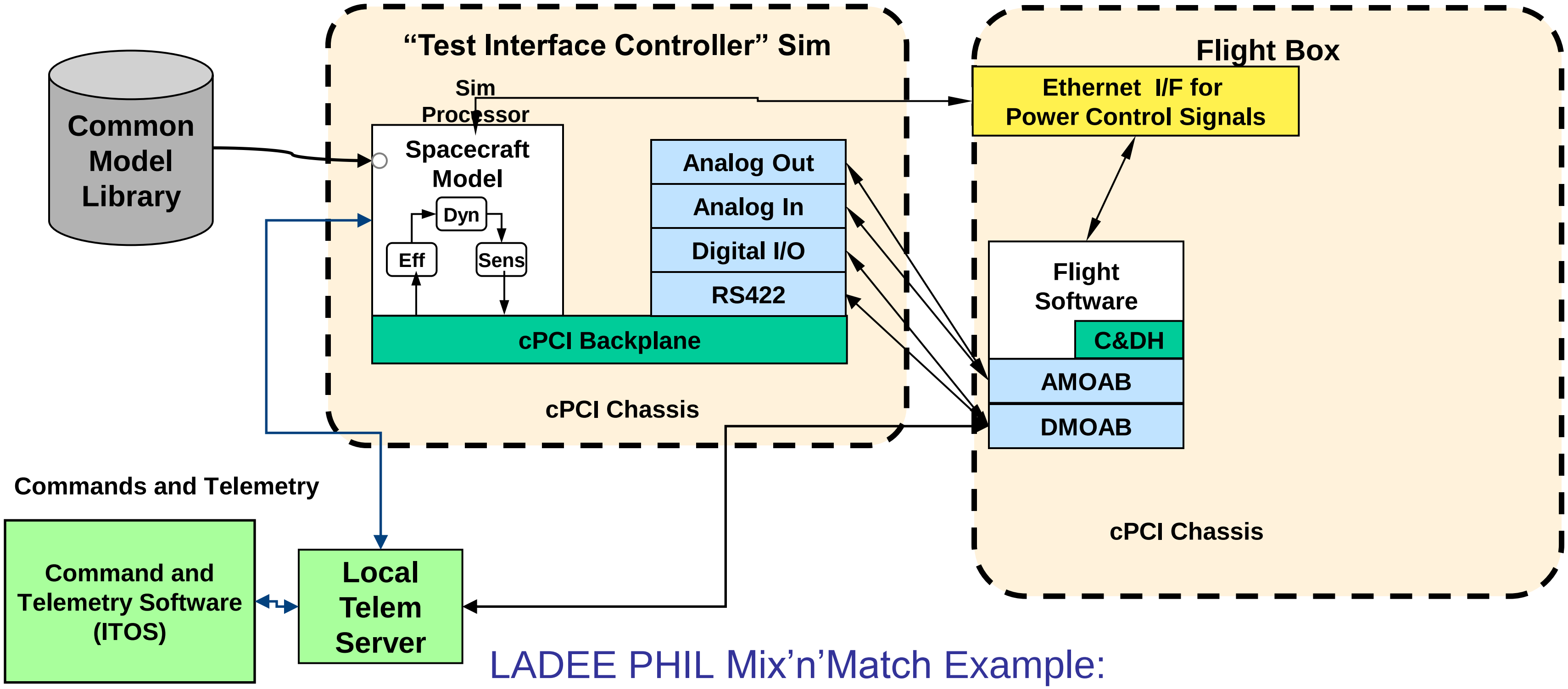
Processor-in-the-Loop (PIL) Simulation



- Models auto-coded and running on RT processors
- Inexpensive "flight-like" processor
- Tests auto-coding process & integration with C&DH software
- Integration with Telemetry Software allows early development/testing of downlink
- Can be used for initial code size and resource utilization analysis
- Software Development Testbed for Application layer (cFE Apps)
  - Includes Device Virtualization Apps (Hardware I/O and Protocol)
  - Includes vxWorks instantiation of OSAL ; but not Flight H/W Device Drivers



- LADEE HIL (Hardware-in-the-Loop) Simulation
  - Flight code runs on Flight Avionics EDU
  - Provides testing of FSW with Avionics I/O
  - Definitive answers on resource utilization
  - Highest fidelity simulations for verification/validation
  - Testbed for all layers including Flight Hardware Device Drivers



LADEE PHIL Mix'n'Match Example:

- **Initially only DMOAB (RS422) and AMOAB Available**
  - **No Power Boards in Avionics Box**
- **Used Ethernet for other H/W interfaces**
- **Re-configured during runtime initialization**
  - **Drivers for missing H/W replaced with Ethernet Drivers**
- **Application Code unchanged**

Flexible Infrastructure with Evolving Layers and Test-Beds meets the Challenge

- **CAST Re-use (Proven Standards and tools)**
  - Posix File I/O, vxWorks, CCSDS Space Packets
  - SpaceWire, Ethernet, RS422 / SPI
  - django/python Database interface-generation Framework
  - Simulink / Matlab code generation
- **"Test Like You Fly" / "Integrate Early and Often"**
  - Flight Ground Software and Database used as GSE for early Device Integration
    - **Minimize redundant tlm/cmd work (via SCIMI)**
  - Common Code used in all Closed-Loop Test Infrastructure
    - **Evolving Fidelity: WSIM, PIL, HIL**
- **Efficient Layering**
  - Device Virtualization – HWIO Apps
    - **Posix File I/O**
  - Loosely Coupled FSW Components
    - **cFE / CFS**
    - **Publish / Subscribe**
  - Virtual Processors